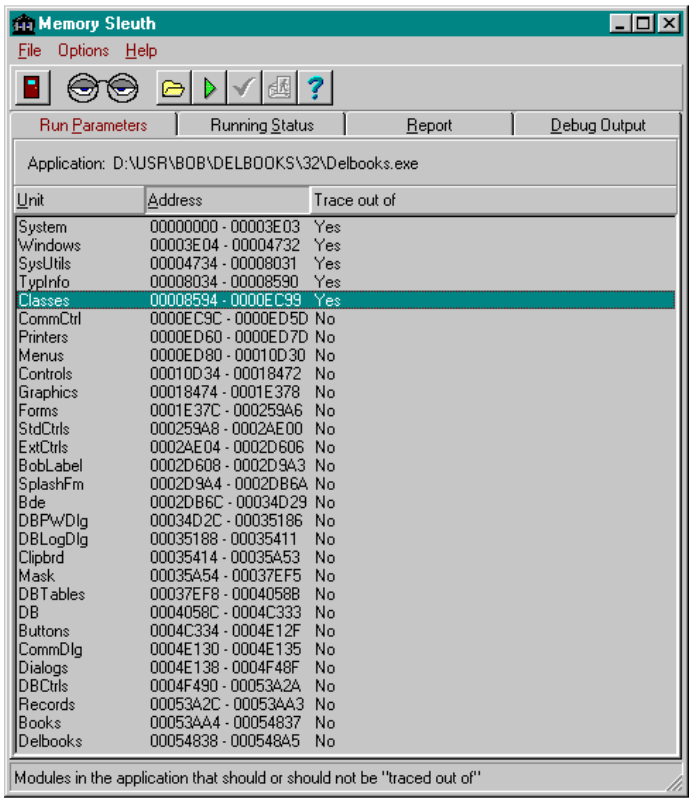
Review: Memory Sleuth 1.0

Reviewed by Bob Swart

You've written your great new Delphi 2 application and now it's time to deliver. And on the eve of delivery, you find out the program is leaking memory and, worse, resources. What to do? Delivery is due tomorrow morning! Should we order an extra large pizza (with lots of pepperoni)? Why not call in the professional help of... Memory Sleuth – after all, its only job is to detect memory and resource leaks in applications. And we'll soon see it's the *best* at what it does!



➤ *Figure 1*

Shadowing

Like a true detective, Memory Sleuth works by shadowing the offender. In this case any 32-bit application written in Delphi 2.0x that is compiled with full debug information included in the executable (both the `{$D+,L+}` flags and the `debug info in exe` option are needed). Now start up Memory Sleuth and open the offending application, like the 32-bit version of DELBOOKS.EXE (Figure 1).

Memory Sleuth will show the address of the units which make up the application. Note that a few units are specified *not to trace out of*. This means that we won't see any error reports for leakages in these specific units (such as the `system` unit), but rather for the units that call them. We don't want the little kid who done it, we want to catch the boss who called for it! Setting `trace out of` to `True` for a lot of VCL units will help identify the spot in our own components and application.

The user interface of Memory Sleuth is simplicity itself: the manual needs only a few pages to explain the options and features since most of the job is done "undercover" while the application being tested is running.

The Scene of the Crime

A simple click on the `Run` button is needed to start the application we want to test. After working a while with the test application we can switch over to the `Running Status` tab in Memory Sleuth to view the offender in action at the scene of the crime! See Figure 2.

This page shows us memory usage plus `USER` and `GDI` resources, including peaks (so you can even use Memory Sleuth to determine the maximum memory and resources needed if you just walk through every screen, dialog and option of your program).

Messages

Memory Sleuth is also able to eavesdrop on debug messages that are being sent by the application using either the `OutputDebugString` API or the Memory Sleuth Debugging Driver itself. We can use this for extended debugging purposes as well (or in case you don't have a copy of the 32-bits version of Turbo Debugger, for example).

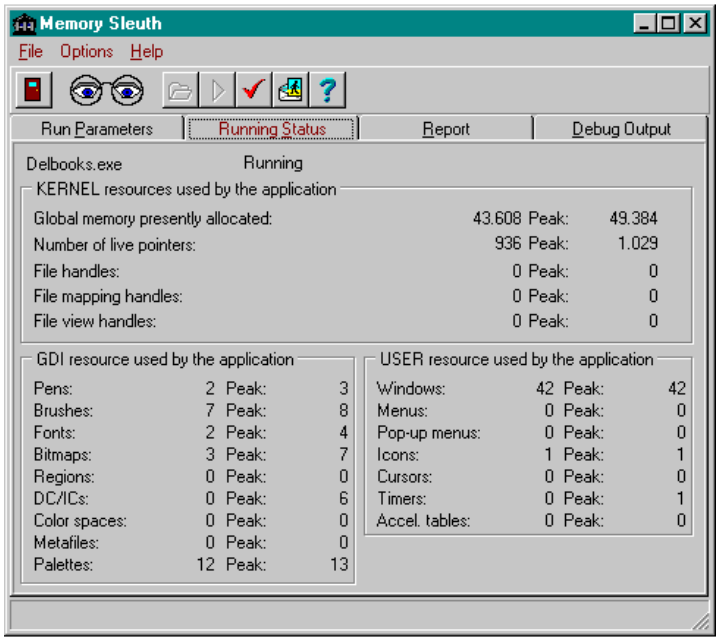
Memory Sleuth also catches accidental heap memory overwrites. By turning on overwrite protection, Memory Sleuth will watch to make sure you never write beyond the bounds of an allocated memory block. This is done by appending a few bytes with a known value and re-checking them later. Any overwrite will be detected.

Evidence

Unlike peak memory and resource and heap overwrites, real memory and resources leakages will of course only be detectable after the program has terminated. Closing the DELBOOKS application and returning to Memory Sleuth, the devastating results are shown in the screenshot on the cover.

It looks like we've got a lot of problems: all kinds of memory and resource leaks! I'm really pretty sure I didn't leave any leaks myself (whatever is allocated sure is freed as well). That's where the extra facts come in: Memory Sleuth comes with a text file and hard evidence that exposes several leaks in 8 units of Delphi 2's VCL. So, what we're seeing here is in fact not

➤ *Figure 2*



DELBOOKS leaking memory, but the VCL.

So, the first thing any Memory Sleuth user needs to do is fix the VCL according to the supplied file VCL.TXT. If you don't do this you won't be able to distinguish between leaks in your application and those in the VCL.

One of the units (`SysUtils`) can't be re-compiled without Turbo Assembler. Two of the fixes were also done (one in a different way) by Borland for Delphi 2.01. After I fixed the VCL and re-ran Memory Sleuth, all the leaks disappeared! *[Take a Gold Star for tidy programming, Dr.Bob! Editor].*

Case Closed

Having seen Memory Sleuth in action, I can't imagine ever releasing a Delphi 2 application again without having checked it with Memory Sleuth. It takes just a few minutes and the results are sure worth it!

Detective Fee

Like a good detective, Memory Sleuth is excellent value, considering the heartache (and customer complaints) it can save you. For US$49 you can buy it direct from TurboPower Software, or from your favourite software tools retailer. The VCL.TXT file alone is worth half the price!

Please note again that Memory Sleuth only supports 32-bit applications developed with Borland Delphi 2. DLLs need to be turned into executables to be traceable, but that is usually not a big problem. For Delphi 1 applications check out MEMMOND, Memory Sleuth's 16-bit little brother (also written by Per Larsen).

TurboPower can be contacted by telephone on +1 719 260 9136, or fax +1 719 260 7151, or GO TURBOPOWER on CompuServe, or visit http://www.tpower.com